# CWE Detail – CWE-1255

## Description

A device's real time power consumption may be monitored during security token evaluation and the information gleaned may be used to determine the value of the reference token.

## Extended Description

The power consumed by a device may be instrumented and monitored in real time. If the algorithm for evaluating security tokens is not sufficiently robust, the power consumption may vary by token entry comparison against the reference value. Further, if retries are unlimited, the power difference between a "good" entry and a "bad" entry may be observed and used to determine whether each entry itself is correct thereby allowing unauthorized parties to calculate the reference value.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2020-12788: CMAC verification vulnerable to timing and power attacks.

## Related Attack Patterns (CAPEC)

* CAPEC-189

## Modes of Introduction

**•** Architecture and Design: The design of the algorithm itself may intrinsically allow the power side channel attack to be effective

**•** Implementation: This weakness may be introduced during implementation despite a robust design that otherwise prevents exploitation

## Common Consequences

**•** Impact: Modify Memory, Read Memory, Read Files or Directories, Modify Files or Directories, Execute Unauthorized Code or Commands, Gain Privileges or Assume Identity, Bypass Protection Mechanism, Read Application Data, Modify Application Data, Hide Activities — Notes: As compromising a security token may result in complete system control, the impacts are relatively universal.

## Potential Mitigations

**•** Architecture and Design: The design phase must consider each check of a security token against a standard and the amount of power consumed during the check of a good token versus a bad token. The alternative is an all at once check where a retry counter is incremented PRIOR to the check. (Effectiveness: N/A)

**•** Architecture and Design: Another potential mitigation is to parallelize shifting of secret data (see example 2 below). Note that the wider the bus the more effective the result. (Effectiveness: N/A)

**•** Architecture and Design: An additional potential mitigation is to add random data to each crypto operation then subtract it out afterwards. This is highly effective but costly in performance, area, and power consumption. It also requires a random number generator. (Effectiveness: N/A)

**•** Implementation: If the architecture is unable to prevent the attack, using filtering components may reduce the ability to implement an attack, however, consideration must be given to the physical removal of the filter elements. (Effectiveness: N/A)

**•** Integration: During integration, avoid use of a single secret for an extended period (e.g. frequent key updates). This limits the amount of data compromised but at the cost of complexity of use. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** Since the algorithm uses a different number of 1's and 0's for password validation, a different amount of power is consumed for the good byte versus the bad byte comparison. Using this information, an attacker may be able to guess the correct password for that byte-by-byte iteration with several repeated attempts by stopping the password evaluation before it completes.

**•** This code demonstrates the transfer of a secret key using a Parallel-In/Parallel-Out shift. In a parallel shift, data confounded by multiple bits of the key, not just one.